Genetic Algorithms for the Traveling Salesman Problem

Conference Paper · January 1985

CITATIONS
518

READS
3,480

4 authors, including:

John J. Grefenstette
University of Pittsburgh
161 PUBLICATIONS **11,284** CITATIONS
SEE PROFILE

Rajeev Gopal
Hughes Network Systems, LLC
22 PUBLICATIONS **597** CITATIONS
SEE PROFILE

Dirk Van Gucht
Indiana University Bloomington
195 PUBLICATIONS **4,039** CITATIONS
SEE PROFILE

Genetic Algorithms for the Traveling Salesman Problem

John Grefenstette[1], Rajeev Gopal,
Brian Rosmaita, Dirk Van Gucht

Computer Science Department
Vanderbilt University

Abstract

This paper presents some approaches to the application of Genetic Algorithms to the Traveling Salesman Problem. A number of representation issues are discussed along with several recombination operators. Some preliminary analysis of the Adjacency List representation is presented, as well as some promising experimental results.

1. Introduction

Genetic Algorithms (GA's) have been applied to a variety of function optimization problems, and have been shown to be highly effective in searching large, complex response surfaces even in the presence of difficulties such as high-dimensionality, multimodality, discontinuity and noise [4]. However, GA's have not been applied extensively to combinatorial problems. The major obstacle is in finding an appropriate representation. This paper presents some approaches to the design of GA's for a well known combinatorial optimization problem -- the Traveling Salesman Problem (TSP). The TSP is easily stated: Given a complete graph with N nodes, find the shortest Hamiltonian path through the graph. (In this paper, we will assume Euclidean distances between nodes.) The TSP is NP-Hard, which probably means that any algorithm which computes an exact solution of the TSP requires an amount of computation time which is exponential in N, the size of the problem [5]. In addition to its many important applications, the TSP is often used to illustrate heuristic search methods [2,7,8], so it is natural to investigate the use of GA's for this problem.

Choosing an appropriate representation is the first step in applying GA's to any optimization problem. If the problem involves searching an N-dimensional space, the representation problem is often solved by allocating a sufficient number of bits to each dimension to achieve the desired accuracy. For the TSP, the search space is a space of permutations and the representation problem is more complex. Consider a path representation in which a tour is represented by a list of cities: (a b c d e f). The first problem is that the representation is not unique: each tour has N representations This can be solved by fixing the initial city. Another problem is that the crossover operator does not generally yield offspring which are legal tours. For example, suppose we cross tours (a b c d e) and (a d e c b) between the third and fourth cities. We get as offspring (a b c c b) and (a d e d e), neither of which are legal tours. Finally, there is a problem in applying the hyperplane analysis of GA's to this representation. The definition of a hyperplane is unclear in this representation. For example, (a # # # #) appears to be a first order hyperplane, but it contains the entire space. The problem is that in this representation, the semantics of an allele in a given position depends on the surrounding alleles. Intuitively, we hope that GA's will tend to construct good solutions by identifying good building blocks and eventually combining these to get larger building blocks. For the TSP, the basic building blocks are edges. Larger building blocks correspond to larger subtours. The path representation does not lend itself to the description of edges and longer subtours in ways which are useful to the GA.

In section 2, we present two representations which offer some improvements over the path representation. Section 3 discusses the design of a heuristic recombination operator for what we consider to be the most promising representation. In section 4, some preliminary experimental

[1]Research supported in part by the National Science Foundation under Grant MCS-8305693.

results are described for the TSP. Section 5 discusses some future directions.

2. Representations for TSP

2.1. Ordinal Representation

In the ordinal representation, a tour is described by a list of N integers in which the ith element can range from 1 to (N-i+1). Given a path representation of a tour, we can construct the ordinal representation TourList as follows: Let FreeList be an ordered list of the cities. For each city in the tour, append the position of that city in the FreeList to the TourList and delete that city from the FreeList. For example, the path tour (a c e d b) corresponds to an ordinal tour (1 2 3 2 1) as shown:

TourList	FreeList
()	(a b c d e)
(1)	(b c d e)
(1 2)	(b d e)
(1 2 3)	(b d)
(1 2 3 2)	(b)
(1 2 3 2 1)	()

Note that it is necessary to fix the starting city to avoid multiple representation of tours.

A similar procedure provides a mapping from the ordinal representation back to the path representation. In fact, the mapping between the two representations is one-to-one.

The primary advantage of the ordinal representation is that the classical crossover operator may be freely applied to the ordinal representation and will always produce the ordinal representation of a legal tour. However, the results of crossover may not bear much relation to the parents when translated to the path representation. For example, consider the following two tours:

ordinal tours	path tours
(1 2 3 2 1)	(a c e d b)
(2 4 1 1 1)	(b e a c d)

Suppose that we cross the ordinal tours between the second and third positions. We get the following tours as offspring:

ordinal tours	path tours
(1 2 1 1 1)	(a c b d e)
(2 4 3 2 1)	(b e d c a)

The subtours corresponding to the genes in the ordinal tours to the left of the crossover point do not change. However, the subtours corresponding to genes to the right of the crossover points are disrupted in a fairly random way. Furthermore, the closer the crossover point is to the front of the tour, the greater the disruption of subtours in the offspring.

As predicted by the above consideration of subtour disruptions, experimental results using the ordinal representation have been generally poor. In most cases, a GA using the ordinal representation does no better than random search on the TSP.

2.2. Adjacency Representation

In the adjacency representation, a tour is described by a list of cities. There is an edge in the tour from city i to city j iff the allele in position i is j. For example, the path tour (1 3 5 4 2) corresponds to the adjacency tour (3 1 5 2 4). Note that any tour has exactly one adjacency list representation.

2.2.1. Crossover Operators

Unlike the ordinal representation, the adjacency representation does not allow the classical crossover operator. Several modified crossover operators can be defined.

Alternating Edges

Using the alternating edges operator, an offspring is constructed from two parent tours as follows: First choose an edge at random from one parent. Then extend the partial tour by choosing the appropriate edge from the other parent.

Continue extending the tour by choosing edges from alternating parents. If the parent's edge would introduce a cycle into a partial tour, then extend the partial tour by a random edge which does not introduce a cycle. Continue until a complete tour is constructed.

For example, suppose we have

mom = (2 3 4 5 6 1)
dad = (2 5 1 6 4 3)

Then we might get the following offspring:

kid = (2 5 4 1 6 3)

where the only random edge introduced into the offspring is the edge (4 1). All other edges were inherited by alternately choosing edges from parents, starting with the edge (1 2) from mom.

Experimental results with the alternating edges operator have been uniformly discouraging. The obvious explanation seems to be that good subtours are often disrupted by the crossover operator. Ideally, an operator ought to promote the development of coadapted alleles, or in the TSP, longer and lónger high performance subtours. The next operator was motivated by the desire to preserve longer parental subtours.

Subtour Chunks

Using the subtour chunking operator, an offspring is constructed from two parent tours as follows: First choose a subtour of random length from one parent. Then extend the partial tour by choosing a subtour of random length from the other parent. Continue extending the tour by choosing subtours from alternating parents. During the selection of a subtour from a parent, if the parent's edge would introduce a cycle into a partial tour, then extend the partial tour by a random edge which does not introduce a cycle. Continue until a complete tour is constructed.

Subtour chunking performed better than alternating edges, as expected, but the absolute performance was still unimpressive. An analysis of the allocation of trials to hyperplanes provide a partial explanation for the poor performance of this operator.

2.2.2. Hyperplane Analysis

The primary advantage of the adjacency representation is that it permits the kind of hyperplane analysis which has been applied to the N-dimensional function optimization GA paradigm [1,3,6]. Hyperplanes defined in terms of a single defining position correspond to the natural building blocks, i.e., edges, for the TSP problem. For example, the hyperplane (# # # 2 #) is the set of all permutations in which the edge (4 2) occurs. We briefly summarize the main points of the classical hyperplane analysis of GA's: In the absence of recombination operators, selection of structures for reproduction in proportion to the structure's observed relative performance allocates trials to all represented hyperplanes in the population (roughly) according to the following formula:

$$M(H,t+1) = M(H,t)*(u(H,t) / u(P,t))$$

where

M(H,t) = # of representatives of H at time t

u(H,t) = observed performance of H at time t

u(P,t) = mean performance of population at time t.

The elements of any hyperplane partition compete against the other elements of that partition, with the better performing elements eventually propagating through the population. This in turn leads to a reduction in the dimensionality of the search space, and the construction of larger high performance building blocks.

In the adjacency representation, a first order hyperplane partition consists of all of the hyperplanes which are defined on the same position. For example:

{ (# # # 1 #), (# # # 2 #), (# # # 3 #), (# # # 5 #) }

is a first order hyperplane partition. Each element of the partition contains an equal

number of tours. Selection is supposed to distinguish among the elements of this partition and to favor the high performance hyperplanes. However, the following theorem shows that selection has very little information on which to allocate trials to competing first order hyperplanes.

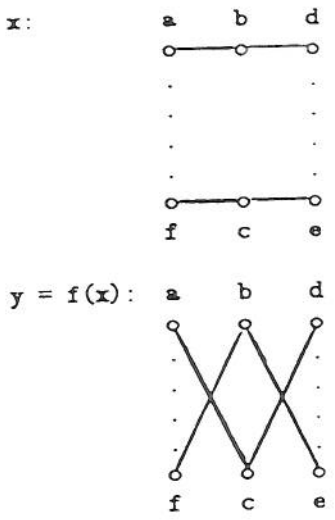
Theorem 1. Suppose that H_{ab} and H_{ac} are two first-order hyperplanes defined by the edges (a b) and (a c), respectively, in a Euclidean TSP. Then $| u(H_{ab}) - u(H_{ac}) | \leq 4(ab + ac)$ where ab and ac represent the lengths of the edges (a b) and (a c), respectively.

Proof. We show that there is a one-to-one mapping f between the tours in H_{ab} and the tours H_{ac} such that if x is a tour in H_{ab} and $y = f(x)$ is the corresponding tour in H_{ac}, then

$$| \text{Length}(y) - \text{Length}(x) | \leq 4(ab+ac).$$

The theorem follows directly.

The following illustrates the mapping f:



That is, y is obtained by exchanging the nodes b and c in the tour x. Using the triangle inequality, it is easy to show that:

$$-(4ab + 2ac) \leq \text{Distance}(y) - \text{Distance}(x) \leq (4ac + 2ab).$$

So

$$| \text{Distance}(y) - \text{Distance}(x) | \leq 4(ab+ac).$$

QED.

In practice, the observed difference between competing first order hyperplanes is usually an order of magnitude less than the bounds in the theorem. And since the overall tour length is generally very large compared to the bound in the theorem, there is generally no significant difference between the mean relative performance of any two competing first order hyperplanes. Our experimental studies have shown that the difference in the observed performance of competing first order hyperplanes in a TSP of size 20 is generally less than 5% of the mean population tour length. In larger problems, this difference can be expected to rapidly approach zero.

One might suspect that the TSP is not a suitable problem for GA's, that the TSP is in some sense GA-Hard. Bethke[1] characterizes some problems for which GA's are unsuitable. Informally, Bethke shows that there are functions and representations for which the low order hyperplanes can mislead the GA into allocating trials to suboptimal areas of the search space. However, Bethke's techniques, which involve the Walsh transform of the objective function, apply to one-dimensional functions of a real variable using a fixed-point representation. A similar set of results may be derivable for combinatorial problems using the adjacency representation. But Theorem 1 does not indicate that the information in the first order hyperplanes of the adjacency representation is misleading, just that it is buried. In other words, measuring the fitness of a tour by the tour length may be too crude a measure for apportioning credit. We now describe a crossover operator which performs a secondary apportionment of credit at the level of individual alleles.

3. Heuristic Crossover

Theorem 1 shows that selection alone may not be able to properly allocate trials to first order hyperplanes, given our adjacency representation for the TSP. The heuristic crossover operator attempts to perform a secondary apportionment

of credit at the allele level. This operator constructs an offspring from two parent tours as follows: Pick a random city as the starting point for the child's tour. Compare the two edges leaving the starting city in the parents and choose the shorter edge. Continue to extend the partial tour by choosing the shorter of the two edges in the parents which extend the tour. If the shorter parental edge would introduce a cycle into the partial tour, then extend the tour by a random edge. Continue until a complete tour is generated.

In order to compare this operator with the previous two recombination operators, 1000 random pairs of parents were chosen for a TSP of size 20. For each pair of parents, an offspring was constructed according to each of the crossover operators. For all three operators, the offspring generally inherited about 30% of the edges from each parent. The remaining 40% were random edges introduced by the recombination operator to create a legal tour. For the first two operators, the offspring generally show no improvement in overall tour length when compared to the better parent. Not surprisingly, the heuristic crossover produces offspring which are, on average, about 10% better than the better parent. It seems reasonable that such an improvement should give selection a way to promote the propagation of good edges through the population. The next section shows some experimental results which confirm this expectation.

It is important to note that, with the proper choice of data structures, the heuristic crossover operator can be implemented to run as a linear function of the length of the structures [9]. This implies that, if E is the number of trials and N is the number of cities, our GA's for the TSP run with asymptotic complexity O(EN), the same as pure random search.

4. Experimental Results

This section describes some experiments with the adjacency representation and the heuristic crossover operator. For each experiment, N cities were randomly placed in a square Euclidean space. The initial population consisted of randomly generated tours. The selection method was based on the expected value model. The crossover rate was set at 50%, and there was no explicit mutation operator.

Figure 1 shows the results of a 50 city problem, Figure 2 shows a 100 city problem and Figure 3 shows a 200 city problem. Each Figure shows a representative tour from the initial population, the best tour obtained part way through the search, and the best tour obtained after the entire search, along with a randomly selected tour in the final population. It can be seen, especially in Figues 2 and 3, that good subtours tend to survive and to propagate. The figures also show that there is still a good deal of diversity in the final population.

Statistical techniques [2] allow us to estimate that the expected length of an optimal tour for experiment 1 is approximately 37.45. The optimal tour obtained by the GA differs from this expected optimum by about 25%. After an equal number of trials, random search produces a best tour of length 148.6, nearly 300% longer than the optimal tour. The optimal tour obtained in experiment 2 differs from the expected optimum by 16%. The optimal tour obtained in experiment 3 differs from the expected optimum by about 27%. These results are encouraging and suggest that further investigation of this approach is warranted.

Experiments show that GA's which use heuristic crossover but not selection perform better than random search but significantly worse than GA's which use both selection and heuristic crossover. That is, there appears to be a symbiotic relationship between the two levels of credit assignment performed by selection and heuristic crossover. We are currently working on clarifying the relationship between selection and the heuristic crossover operator.

5. Future Directions

This papers presents some preliminary observations and experiments. Many more questions about the TSP need to be investigated. Some interesting future projects include:

Combining GA's with other heuristics. In may be useful to heuristically choose the initial

population of tours. For example, the nearest neighbor algorithm can generate a set of relatively good tours when started from various initial cities. For very large problems, nearest neighbor can be approximated by choosing a random set of cities and taking the one closest to the current city. Heuristics could also be invoked at the end of the GA to do some local modifications to the tours in the final population. For example, the Figures shows many opportunities for improving the final tour by some local edge reversals.

Comparison with simulated annealing. Simulated annealing is another randomized heuristic algorithm which has been applied to very large ($N > 1000$) TSP's. From the published literature on simulated annealing [2,7], it appears that our results are at least competitive. A careful comparison of these two techniques would be very interesting.

Effects of GA parameters. There are several control parameters involved in any GA implementation, such as population size, crossover rate, etc. which may have an effect on the performance of the system. The proposed GA's are sufficiently different from previous GA's that it might be useful to investigate the effects of these parameters for the TSP.

Other combinatorial applications. How do the ideas developed thus far apply to combinatorial problems other than the TSP?

References

1. A. D. Bethke, *Genetic algorithms as function optimizers*, Ph. D. Thesis, Dept. Computer and Communication Sciences, Univ. of Michigan (1981).

2. E. Bonomi and J.-L. Lutton, "The N-city traveling salesman problem: statistical mechanics and the Metropolis Algorithm," *SIAM Review Vol. 26*(4), pp. 551-569 (Oct. 1984).

3. K. A. Dejong, *Analysis of the behavior of a class of genetic adaptive systems*, Ph. D. Thesis, Dept. Computer and Communication Sciences, Univ. of Michigan (1975).

4. K. A. Dejong, "Adaptive system design: a genetic approach," *IEEE Trans. Syst., Man, and Cyber. Vol. SMC-10*(9), pp. 556-574 (Sept 1980).

5. M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman Co., San Fransisco (1979).

6. J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor (1975).

7. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science Vol. 220*(4598), pp. 671-680 (May 1983).

8. J. Pearl, *Heuristics*, Addison-Wesley, Menlo Park (1984).

9. B. J. Rosmaita, *Exodus: An extension of the the genetic algorithm to problems dealing with permutations*, M.S. Thesis, Computer Science Department, Vanderbilt University (Aug. 1985).

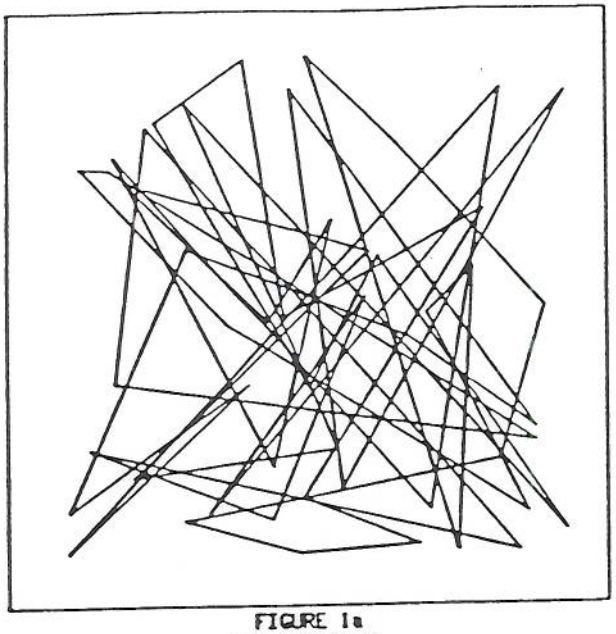


FIGURE 1a
50 CITIES
DISTANCE = 197.82
INITIAL POPULATION

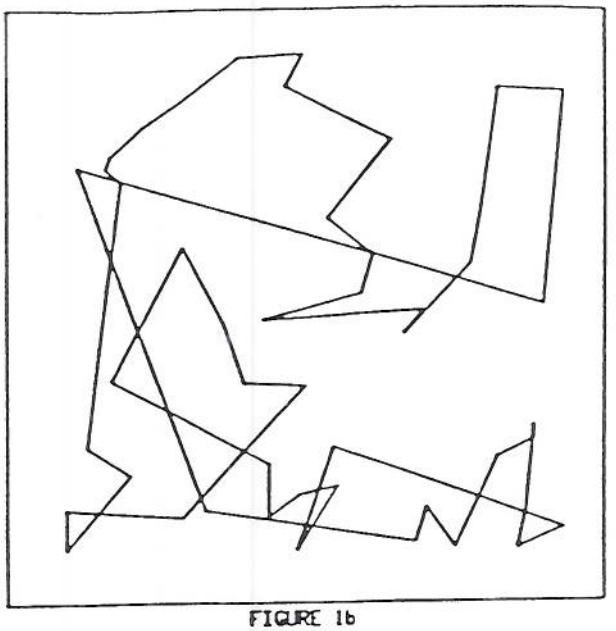


FIGURE 1b
50 CITIES
DISTANCE = 64.76
GENERATION 38 1969 TRIALS

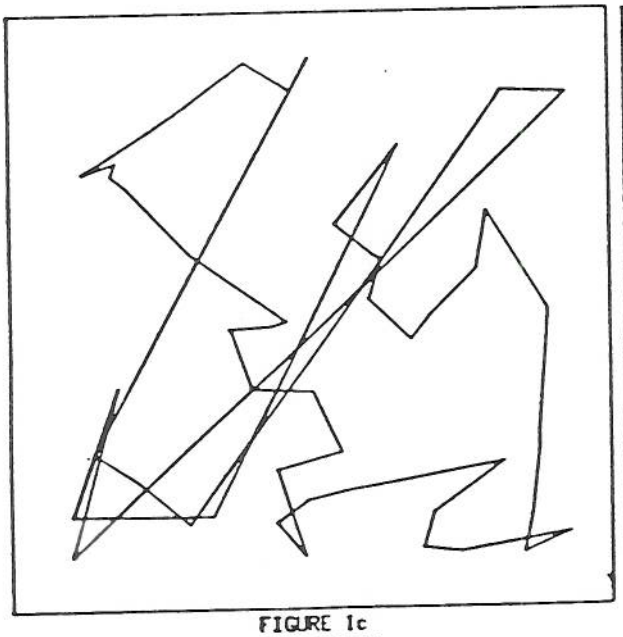


FIGURE 1c
50 CITIES
DISTANCE = 68.32
FINAL POPULATION

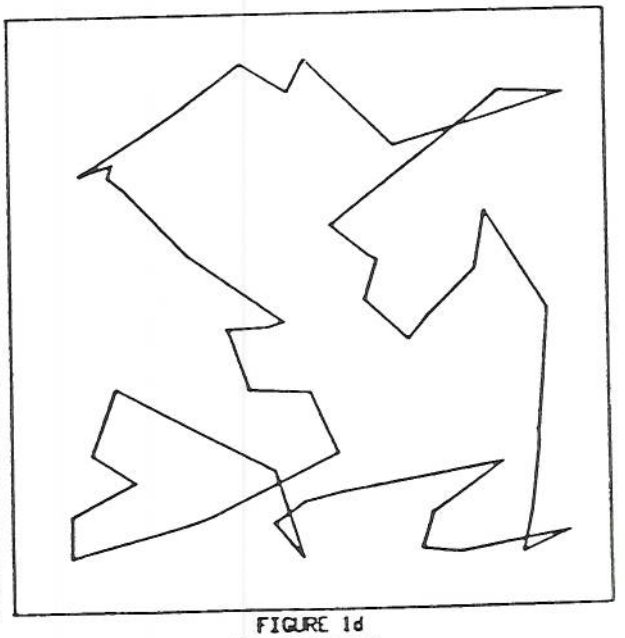


FIGURE 1d
50 CITIES
DISTANCE = 46.84
GENERATION 294 14686 TRIALS

Figure 1.

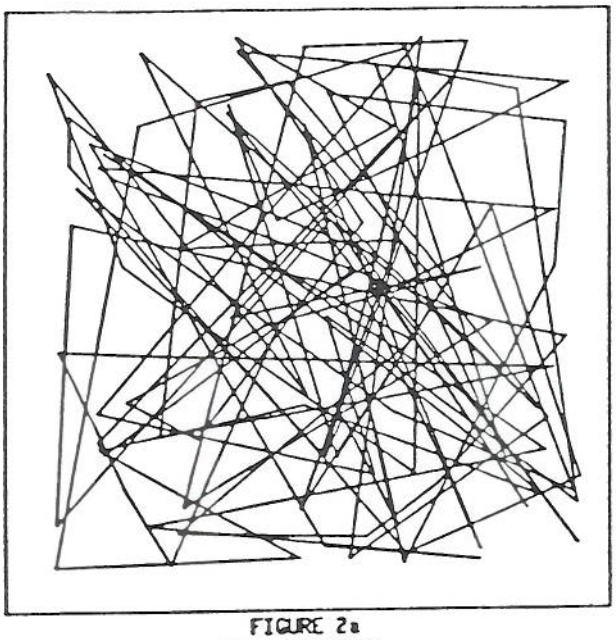


FIGURE 2a
100 CITIES
DISTANCE = 547.12
INITIAL POPULATION

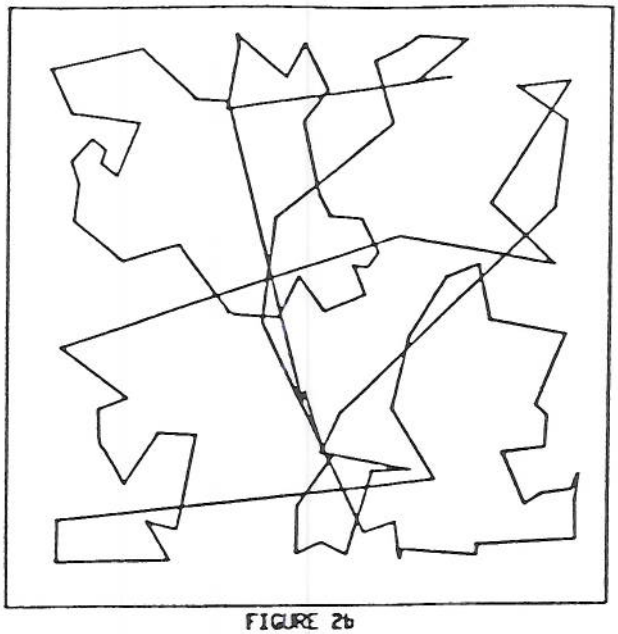


FIGURE 2b
100 CITIES
DISTANCE = 118.47
GENERATION 125 6296 TRIALS

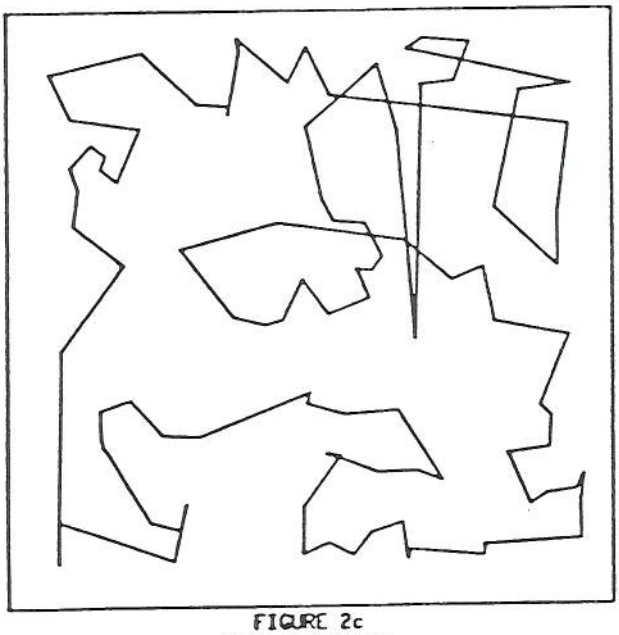


FIGURE 2c
100 CITIES
DISTANCE = 99.84
FINAL POPULATION

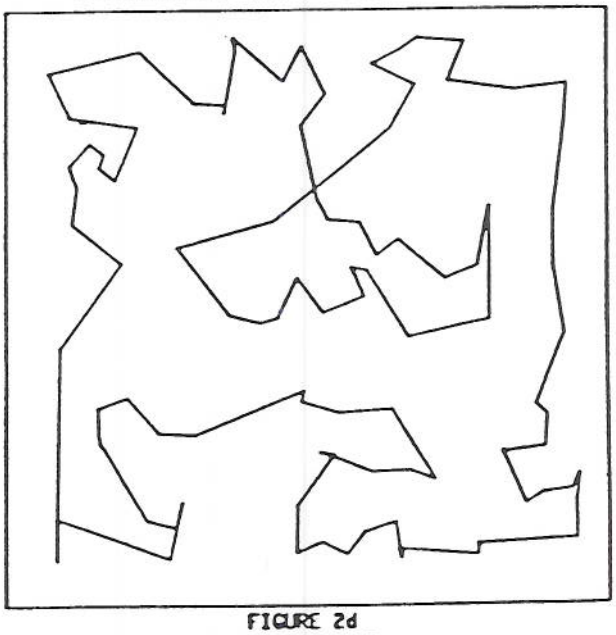


FIGURE 2d
100 CITIES
DISTANCE = 87.21
GENERATION 407 20330 TRIALS

Figure 2.

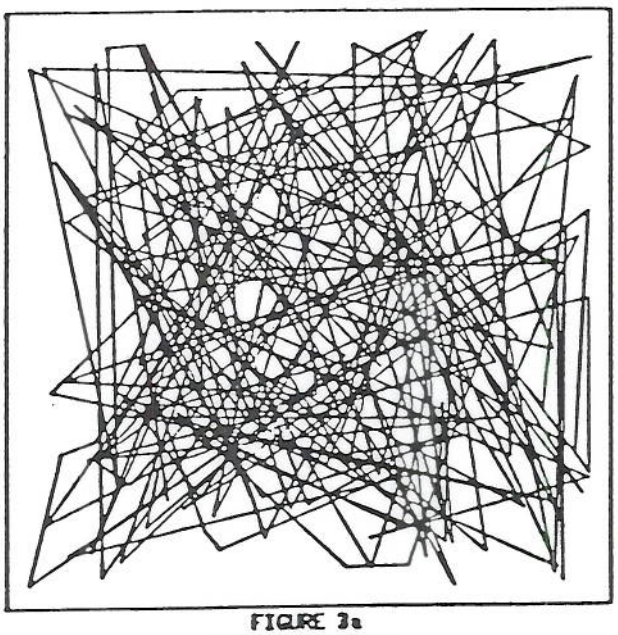


FIGURE 3a
200 CITIES
DISTANCE = 1475.68
INITIAL POPULATION

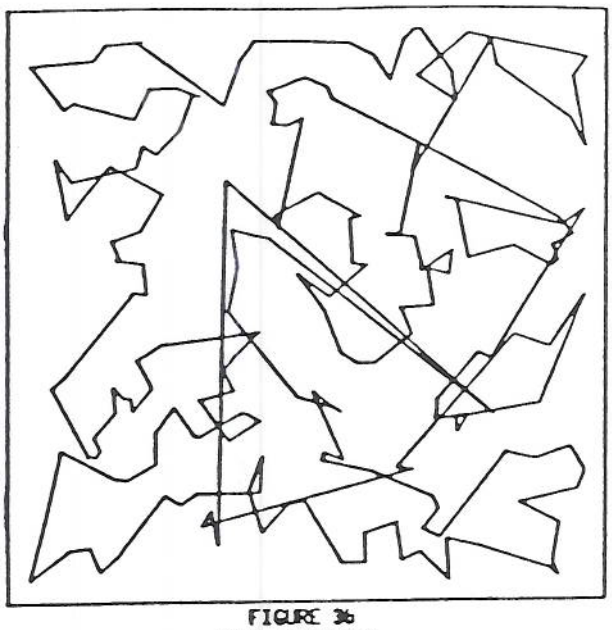


FIGURE 3b
200 CITIES
DISTANCE = 223.81
GENERATION 227 11373 TRIALS

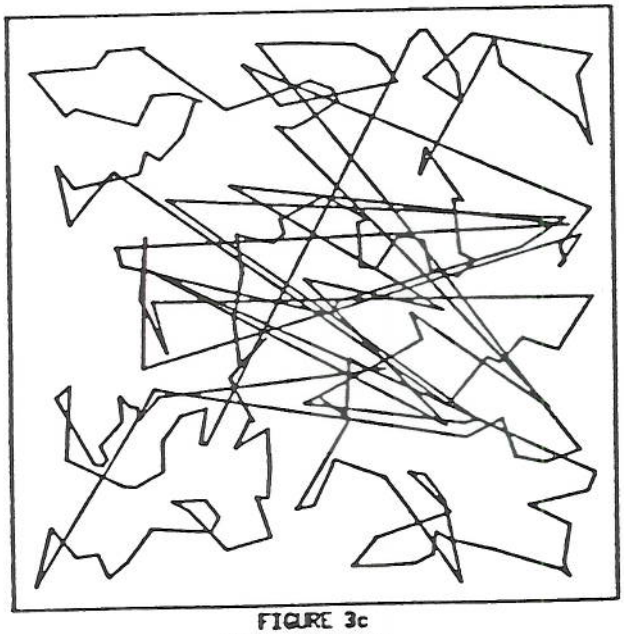


FIGURE 3c
200 CITIES
DISTANCE = 351.22
FINAL POPULATION

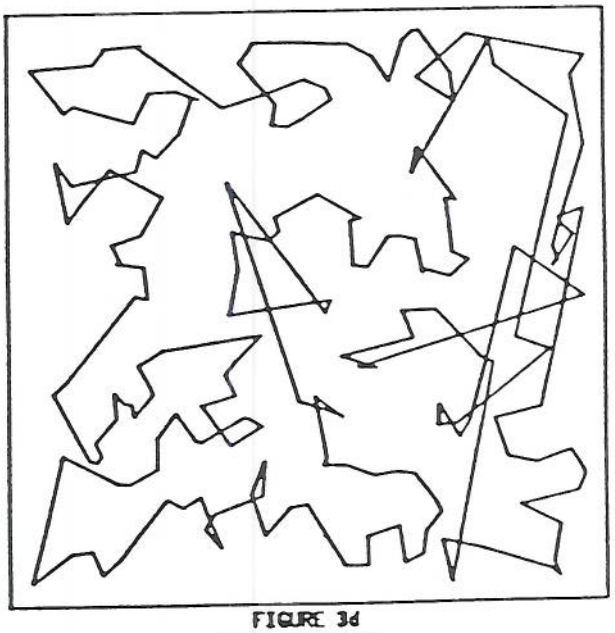


FIGURE 3d
200 CITIES
DISTANCE = 203.46
GENERATION 493 24596 TRIALS

Figure 3.